chapter 6

Applied Cryptography

There are two types of encryption: one that will prevent your sister from reading your diary and one that will prevent your government.

—Bruce Schneier

In this chapter, you will learn how to

- Correctly use the elements of cryptography
- Examine cipher suites and common uses
- Identify cryptographic attack methods

None of the still steadily growing Internet commerce would be possible without the use of standards and protocols that provide a common, interoperable environment for exchanging information securely. Due to the wide distribution of Internet users and businesses, the most practical solution to date has been the commercial implementation of public key infrastructures.

Cryptography Use

The use of cryptography grows every day. More and more information becomes digitally encoded and placed online, and all of this data needs to be secured. The best way to do that with current technology is to use encryption. This section considers some of the tasks cryptographic algorithms accomplish and those for which they are best suited. Security is typically defined as a product of five components: confidentiality, integrity, availability, authentication, and nonrepudiation. Encryption addresses all of these components except availability. Key escrow will be one of the most important topics as information becomes universally encrypted; otherwise, everyone may be left with useless data. Digital rights management and intellectual property protection are also places where encryption algorithms are heavily used. Digital signatures combine several algorithms to provide reliable identification in a digital form.

Confidentiality

Confidentiality typically comes to mind when the term *security* is brought up. Confidentiality is the ability to keep some piece of data a secret. In the digital world, encryption excels at providing confidentiality. In most cases, symmetric encryption is favored because of its speed and because some asymmetric algorithms can significantly increase the size of the object being encrypted. Asymmetric cryptography also can be used to protect confidentiality, but its size and speed make it more efficient at protecting the confidentiality of small units for tasks such as electronic key exchange. In all cases, the strength of the algorithms and the length of the keys ensure the secrecy of the data in question.

Integrity

Integrity, better known as **message integrity**, is a crucial component of message security. When a message is sent, both the sender and recipient need to know that the message was not altered in transmission. This is especially important for legal contracts—recipients need to know that the contracts have not been altered. Signers also need a way to validate that a contract they sign will not be altered in the future.

Integrity is provided via one-way hash functions and digital signatures. The hash functions compute the message digests, and this guarantees the integrity of the message by allowing easy testing to determine whether any part of the message has been changed. The message now has a computed function (the hash value) to tell the users to resend the message if it was intercepted and interfered with. This hash value is combined with asymmetric cryptography by taking the message's hash value and encrypting it with the user's private key. This lets anyone with the user's public key decrypt the hash and compare it to the locally computed hash, not only ensuring the integrity of the message but positively identifying the sender.

Message integrity has become increasingly important as more commerce is conducted digitally. The ability to independently make sure that a document has not been tampered with is very important to commerce. More importantly, once the document is "signed" with a digital signature, it cannot be refuted that the person in question signed it.

Authentication

Authentication is the matching of a user to an account through previously shared credentials. This information must be protected, and a combination of cryptographic methods is commonly employed. From hashing to key stretching to encryption and digital signatures, multiple techniques are used as part of the operations involved in authentication.

Try This!

Document Integrity

Download a hash calculator that works on your operating system, such as SlavaSoft HashCalc, available at www.slavasoft.com/hashcalc/index.htm. Then create a simple document file with any text you prefer. Save it, and then use the hashing program to generate the hash and save the hash value. Now edit the file, even by simply inserting a single blank space, and resave it. Recalculate the hash and compare.

Tech Tip

HOTP and TOTP

An **HMAC-based One-Time Password (HOTP)** *algorithm is a key component of the Open Authentication Initiative (OATH). YubiKey is a hardware implementation of HOTP that has significant use. Because of how it is implemented, an HOTP offers greater security than simple passwords, as they are no longer valid after use. A Time-based One-Time Password (TOTP) algorithm offers even more protection as it expires after a given interval, whether used or not.*

Nonrepudiation

An item of some confusion, the concept of nonrepudiation is actually fairly simple. Nonrepudiation means that the message sender cannot later deny that they sent the message. This is important in electronic exchanges of data because of the lack of face-to-face meetings. Nonrepudiation is based on public key cryptography and the principle of only you knowing your private key. The presence of a message signed by you, using your private key, which nobody else should know, is an example of nonrepudiation. When a third party can check your signature using your public key, that disproves any claim that you were not the one who actually sent the message. Nonrepudiation is tied to asymmetric cryptography and cannot be implemented with symmetric algorithms.

Digital Signatures

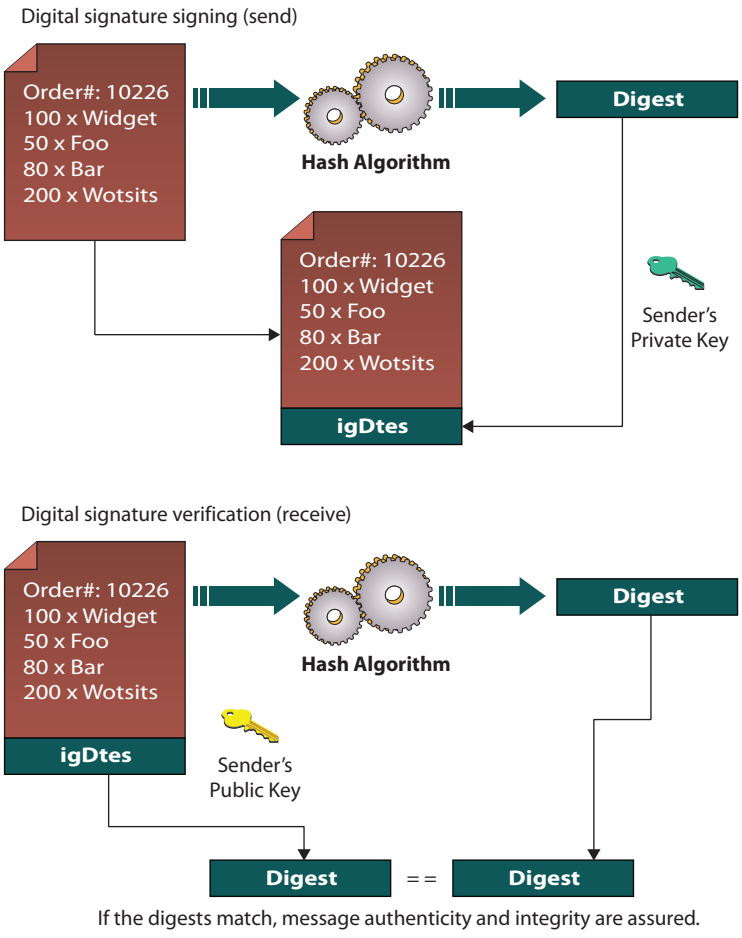
Digital signatures provide a means of verifying the authenticity and integrity of a message: you know both who the sender is and that the message has not been altered. By itself, a digital signature does not protect the contents from unauthorized reading.

Digital signatures have been touted as the key to truly paperless document flow, and they do have promise for improving the system. Digital signatures are based on both hashing functions and asymmetric cryptography. Both encryption methods play an important role in signing digital documents. Unprotected digital documents are very easy for anyone to change. If a document is edited after an individual signs it, it is important that any modification can be detected. To protect against document editing, hashing functions are used to create a digest of the message that is unique and easily reproducible by both parties. This ensures that the message integrity is complete.

A **digital signature** is a cryptographic implementation designed to demonstrate authenticity and identity associated with a message. Using public key cryptography, a digital signature allows traceability to the person signing the message through the use of their private key. The addition of hash codes allows for the assurance of the integrity of the message as well. The operation of a digital signature is a combination of cryptographic

elements to achieve a desired outcome. The steps involved in digital signature generation and use are illustrated in Figure 6.1. The message to be signed is hashed, and the hash is encrypted using the sender's private key. Upon receipt, the recipient can decrypt the hash using the sender's public key. If a subsequent hashing of the message reveals an identical value, two things are known: First, the message has not been altered. Second, the sender possessed the private key of the named sender and therefore is presumably the sender him- or herself.

A digital signature does not by itself protect the contents of the message from interception. The message is still sent in the clear, so if confidentiality of the message is a requirement, additional steps must be taken to secure the message from eavesdropping. This can be done by encrypting the message itself, or by encrypting the channel over which it is transmitted.



• **Figure 6.1** Digital signature operation

Digital Rights Management

Digital rights management (DRM) is the process for protecting intellectual property from unauthorized use. This is a broad area, but the most concentrated focus is on preventing piracy of software or digital content. Before easy access to computers, or the "digital revolution," the content we came in contact with was analog or print based. Although it was possible to copy this content, it was difficult and time consuming to do so, and usually resulted in a loss of quality. It was also much more difficult to send 1000 pages of a handwritten copy of a book to Europe, for example. Computers and the Internet have made such tasks trivial, and now it is very easy to copy a document, music, or video and quickly send it thousands of miles away.

Cryptography has entered the fray as a solution to protect digital rights, though it is currently better known for its failures than its successes. The DVD Content Scramble System (CSS) was an attempt to make DVDs impossible to copy by computer. CSS used an encryption algorithm that was licensed to every DVD player; however, creative programmers were able to retrieve the key to this algorithm by disassembling a software-based DVD player. CSS has been replaced by the Advanced Access Content System (AACS), which is used on the next-generation Blu-ray discs. This system encrypts video content via the symmetric AES algorithm with one or more keys. Several decryption keys have been cracked and released to the Internet, allowing pirates to freely copy the protected content. The music and computer game industries have also attempted several different DRM applications, but nearly all of these have eventually been cracked, allowing piracy.

A common example of DRM that is mostly successful is the broadcast stream of digital satellite TV. Because the signal is beamed from space to every home in North America, the satellite TV provider must be able to protect the signal so that it can charge people to receive it. Smart cards are employed to securely hold the decryption keys that allow access to some or all of the content in the stream. This system has been cracked several times, allowing a subset of users free access to the content; however, the satellite TV providers learned from their early mistakes and upgraded to new smart cards to correct the old problems.

DRM will also become very important in the industry of Software as a Service (SaaS). Similar to companies that provide satellite TV service, companies that provide SaaS rely on a subscription basis for profitability. If someone could pay for a single license and then distribute that to hundreds of employees, the provider would soon go out of business. Many systems in the past have been cracked because the key was housed inside the software.

Cryptographic Applications

A few applications can be used to encrypt data conveniently on your personal computer. *Pretty Good Privacy (PGP)* is mentioned later in this chapter because it is a useful protocol suite and has historical significance. One of the unique features of PGP is its ability to use both symmetric and asymmetric encryption methods, accessing the strengths of each method and avoiding the weaknesses of each as well. Symmetric keys are used for bulk encryption, taking advantage of the speed and efficiency of symmetric encryption. The symmetric keys are passed using asymmetric methods, capitalizing on the flexibility of this method. PGP-based technology is now sold as part of a commercial application, with home and corporate versions.

Filesystem Encryption

Filesystem encryption is becoming a standard means of protecting data while in storage. Even hard drives are available with built-in AES encryption. Microsoft expanded its Encrypting File System (EFS), available since the Windows 2000 operating system, with BitLocker, a boot-sector encryption method that protects data that was introduced with the Windows Vista operating system. BitLocker is also used in Windows Server 2019 and Windows 10 operating systems. BitLocker utilizes AES encryption to encrypt every file on the hard drive automatically. All encryption occurs in the background, and decryption occurs seamlessly when data is requested. The decryption key can be stored in the Trusted Platform Module (TPM) or on a USB key.

Database Encryption

Due partly to increased regulatory concerns and partly to more targeted attacks, databases have begun to offer native support for encryption. Protecting data at rest in the enterprise frequently involves data stored in databases. Building data protection mechanisms into the database systems is not new (it has been around for a long time), but enterprise adoption of this functionality has been slow. Symmetric encryption algorithms such as 3DES and AES are used to encrypt data internally in the database. Protection

mechanisms that can be managed by row and by column are included in most major database applications; the challenge is in convincing organizations to use this proven protection methodology. It does add complexity to the system, but in today's environment of data breaches and corporate espionage, the complexity is easier to manage than the effects of a data loss.

Blockchain

Blockchains are lists of records, where each addition to the list is done by a cryptographic algorithm. While this may seem complex, it serves an important purpose: records in a blockchain are resistant to modification. This permits a distributed ledger that can record transactions and have both verification of additions and protection with respect to integrity. The strength of the integrity comes from both the signing of records and the distributed nature of the blockchain. While a record can be altered, it would require all records after it to also be altered, and thus would be detectable across the copies on the distributed storage of the chain. So, while records are technically alterable, in practicality the system is provably secure.

The concept of blockchains was invented to create the public transaction ledger of cryptocurrencies. A cryptocurrency is a currency system built on a finite set of "rare" numbers that are mined and then "created." As there is no central authority, numbers when mined are entered into the distributed ledger, marking their creation. All transactions are also entered into the ledger, preventing double-spending of the tokens. The use of the distributed public ledger provides the protections that physical tokens provide—only one person can control a given currency token at any given time, and all transactions are immutable.

Blockchain is the record-keeping technology behind Bitcoin. It is a distributed and decentralized public record.

Tech Tip

Public Ledgers

While cryptocurrencies like Bitcoin get the headlines, the true winner in blockchains has been the implementation of distributed public ledgers. Public ledgers *have been used to resolve many challenging problems in tracking financial transactions, supply chains, and other ledger-based issues. Applications for items such as music royalties, DNS systems, and tracking food from farm to table are all examples of blockchain technology solutions under development.*

Use of Proven Technologies

When you're setting up a cryptographic scheme, it is important to use proven technologies. Proven cryptographic libraries and proven cryptographically correct random-number generators are the foundational elements associated with a solid program. Homegrown or custom elements in these areas can greatly increase risk associated with a broken system. Developing your own cryptographic algorithms is beyond the abilities of most groups. Algorithms are complex and difficult to create. Any algorithm that has not had public review can have weaknesses in it. Most good algorithms are approved for use only after a lengthy test and public review phase.

Cipher Suites

In many applications, the use of cryptography occurs as a collection of functions. Different algorithms can be used for authentication, encryption/decryption, digital signatures, and hashing. The term **cipher suite** refers to an arranged group of algorithms. For instance, Transport Layer Security (TLS) has a published TLS Cipher Suite Registry at www.iana.org/assignments/tls-parameters/tls-parameters.xhtml.

Tech Tip

TLS Cipher Suite Example

A cipher suite is the combination of algorithms used during the following stages:

- *Key Agreement*
- *Authentication*
- *Symmetric Cipher and Key Size*
- *Hash Algorithm for Message Authentication*

The choice of algorithms is made by selecting rows from the TLS cipher suite registry during the TLS handshake. It is important to pick algorithms of sufficient strength and avoid older, deprecated ones. The list of cipher suites follows the format shown in the following image.

Cipher ID = 0x00C02B

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS – protocol is TLS
ECDHE – Key agreement - Elliptic Curve Diffie-Hellman Ephemeral
ECDSA – Authentication - Elliptic Curve Digital Signature Algorithm
With – placeholder for readability
AES_128_GCM – Symmetric cipher
SHA256 – Hash for message authentication

Tech Tip

Disabling SSL

Because all versions of SSL, including v3, have exploitable vulnerabilities that make the protocol no longer considered secure, users should not rely on it for security. Google Chrome no longer uses SSL. For Internet Explorer, you need to uncheck the SSL boxes under Internet Options.

Strong vs. Weak Ciphers

There is a wide range of ciphers—some old and some new—each with its own strengths and weaknesses. Over time, new methods and computational capabilities change the viability of ciphers. The concept of strong versus weak ciphers is an acknowledgment that, over time, ciphers can become vulnerable to attacks. The application or selection of ciphers should take into consideration that not all ciphers are still strong. When you're selecting a cipher for use, it is important to make an appropriate choice. For example, if a server offers Secure Sockets Layer (SSL) v3 and TLS, you should choose TLS only, as SSL v3 has been shown to be vulnerable.

As new computing platforms become available, cloud-based GPUs, AI machines, quantum computers, and cryptography will be some of the things that are aimed at by attackers. This has led to a continuation of the cryptographic arms race, where cryptographers are inventing new algorithms, typically specifically designed for either performance considerations, such as lightweight systems, or resistance to mass computational brute force. Significant advancements have been achieved, such as the release of authenticated encryption with associated data (AEAD) systems, TLS v1.3, lightweight cryptographic systems, newer elliptical curves, new hash functions, and new stream ciphers.

Weak/Deprecated Algorithms

Over time, cryptographic algorithms fall to different attacks or just the raw power of computation. The challenge with these algorithms is understanding which ones have fallen to attacks, even though they may still be available for use in software libraries, resulting in their inappropriate application in

use. Although this list will continue to grow, it is important to consider this topic because old habits die hard. Hash algorithms, such as MD5, should be considered inappropriate, as manufactured collisions have been achieved. Even newer hash functions have issues, such as SHA-1, and soon SHA-256. The Data Encryption Standard (DES) and its commonly used stronger form, 3DES, have fallen from favor. The good news is that new forms of these functions are widely available, and in many cases, such as AES, they are computationally efficient, providing better performance.

Secret Algorithms

Algorithms can be broken into two types—those with published details and those where the steps are kept secret. Secrecy has its uses in security. Keeping your password secret, for instance, is an essential element in its proper functionality. Secrecy in how to apply security elements can assist in thwarting reverse engineering. An example of this is the use of multiple rounds of multiple hash functions to provide password security. How many rounds, as well as the order of algorithmic application, is important with respect to the application but is not needed for normal use because it is encoded into the application itself. Keeping this secret can enhance security because it makes reverse engineering difficult, if not impossible.

Secret cryptographic algorithms lead to another issue. Yes, keeping an algorithm secret can impose challenges to those wishing to explore methods of breaking it, but it also reduces the testing of an algorithm by cryptographers attacking it. The most secure algorithms are those that have survived over time the onslaught of cryptographic researchers attacking them.

The best algorithms are always public algorithms that have been published for peer review by other cryptographic and mathematical experts. Publication is important, as any flaws in the system can be revealed by others before actual use of the system. This process greatly encourages the use of proven technologies. Several proprietary algorithms have been reverse-engineered, exposing the confidential data the algorithms try to protect. Examples of this include the decryption of Nikon's proprietary RAW format, white-balance encryption, and the cracking of the ExxonMobil Speedpass RFID encryption. The use of a proprietary system can actually be less secure than using a published system. Whereas proprietary systems are not made available to be tested by potential crackers, public systems are made public for precisely this purpose.

A system that maintains its security after public testing can be reasonably trusted to be secure. A public algorithm can be more secure because good systems rely on the *encryption key* to provide security, not the algorithm itself. The actual steps for encrypting data can be published, because without the key, the protected information cannot be accessed.

One of the most common cryptographic failures is the creation of your own encryption scheme. Rolling your own cryptography, whether in creating algorithms or implementation of existing algorithms yourself, is a recipe for failure. Always use approved algorithms and always use approved crypto libraries to implement them.

Key Exchange

Cryptographic mechanisms use both an algorithm and a key, with the key requiring communication between parties. In symmetric encryption, the secrecy depends on the secrecy of the key, so insecure transport of the key can lead to failure to protect the information encrypted using the key. **Key exchange** is the central foundational element of a secure symmetric

encryption system. Maintaining the secrecy of the symmetric key is the basis of secret communications. In asymmetric systems, the key exchange problem is one of key publication. Because public keys are designed to be shared, the problem is reversed from one of secrecy to one of publicity.

Early key exchanges were performed by trusted couriers. People carried the keys from senders to receivers. One could consider this form of key exchange to be the ultimate in *out-of-band* communication. With the advent of digital methods and some mathematical algorithms, it is possible to pass keys in a secure fashion. This can occur even when all packets are subject to interception. The Diffie-Hellman key exchange is one example of this type of secure key exchange. The Diffie-Hellman key exchange depends on two random numbers, each chosen by one of the parties, and kept secret. Diffie-Hellman key exchanges can be performed *in band*, and even under external observation, as the secret random numbers are never exposed to outside parties.

Tech Tip

Man-in-the-Middle Attack

A man-in-the-middle attack is designed to defeat proper key exchange by intercepting the remote party's key and replacing it with the attacker's key in both directions. If done properly, only the attacker knows that the encrypted traffic is not secure and can be read by the attacker.

Key Escrow

The impressive growth of the use of encryption technology has led to new methods for handling keys. Encryption is adept at hiding all kinds of information, and with privacy and identity protection becoming more of a concern, more information is encrypted. The loss of a key can happen for a multitude of reasons: it might simply be lost, the key holder might be incapacitated or dead, software or hardware might fail, and so on. In many cases, that information is locked up until the cryptography can be broken, and, as you have read, that could be millennia. This has raised the topic of **key escrow**, or keeping a copy of the encryption key with a trusted third party. Theoretically, this third party would only release your key to you or your official designate on the event of your being unable to get the key yourself. However, just as the old saying from Benjamin Franklin goes, "Three may keep a secret if two of them are dead." Any time more than one copy of the key exists, the security of the system is broken. The extent of the insecurity of key escrow is a subject open to debate and will be hotly contested in the years to come.

Tech Tip

Key Escrow Has Benefits and Hazards

Key escrow can solve many of the problems that result when a key is lost or becomes inaccessible, allowing access to data that otherwise would be impossible to access without key escrow, but it can open up private information to unauthorized access.

Key escrow can negatively impact the security provided by encryption, because the government requires a huge, complex infrastructure of systems to hold every escrowed key, and the security of those systems is less efficient than the security of your memorizing the key. However, there are two sides to the key escrow coin. Without a practical way to recover a key if or when it is lost or the key holder dies, for example, some important information will be lost forever. Such issues will affect the design and security of encryption technologies for the foreseeable future.

Session Keys

A **session key** is a symmetric key used for encrypting messages during a communication session. It is generated from random seeds and is used for the duration of a communication session. When correctly generated and propagated during session setup, a session key provides significant levels of protection during the communication session and also can afford perfect forward secrecy. Session keys offer the advantages of symmetric

encryption, speed, strength, simplicity, and, with key exchanges possible via digital methods, significant levels of automated security.

Ephemeral Keys

Ephemeral keys are cryptographic keys that are used only once after they are generated. When an ephemeral key is used as part of the Diffie-Hellman scheme, it forms an Ephemeral Diffie-Hellman (EDH) key exchange. An EDH mechanism generates a temporary key for each connection, never using the same key twice. This provides for perfect forward secrecy. If the Diffie-Hellman scheme involves the use of elliptic curves, it is called Elliptic Curve Diffie-Hellman Ephemeral (ECDHE).

Key Stretching

Key stretching is a mechanism that takes what would be weak keys and "stretches" them to make the system more secure against brute force attacks. A typical methodology used for key stretching involves increasing the computational complexity by adding iterative rounds of computations. To extend a password to a longer length of key, you can run it through multiple rounds of variable-length hashing, each increasing the output by bits over time. This may take hundreds or thousands of rounds, but for single-use computations, the time is not significant. When one wants to use a brute force attack, the increase in computational workload becomes significant when done billions of times, making this form of attack much more expensive.

The common forms of key stretching employed in use today include Password-Based Key Derivation Function 2 (PBKDF2) and Bcrypt.

Key stretching is a mechanism that takes what would be weak keys and "stretches" them. This increases the workload, making the passwords take longer to compute but also much harder to brute force.

PBKDF2

Password-Based Key Derivation Function 2 (PBKDF2) is a key-derivation function designed to produce a key derived from a password. This function uses a password or passphrase and a salt and then applies an HMAC to the input thousands of times. The repetition makes brute force attacks computationally unfeasible.

Bcrypt

Bcrypt is a key-stretching mechanism that uses the Blowfish cipher and salting and then adds an adaptive function to increase the number of iterations. The result is the same as other key-stretching mechanisms (single use is computationally feasible), but when an attempt is made to brute force the function, the billions of attempts make it computationally unfeasible.

Transport Encryption

Transport encryption is used to protect data that is in motion. When data is being transported across a network, it is at risk of interception. An examination of the Open Systems Interconnection (OSI) networking model shows a layer dedicated to transport, and this abstraction can be used to manage end-to-end cryptographic functions for a communication channel. When

Secure Sockets Layer (SSL) was the original transport layer protocol but has since fallen to vulnerabilities and should no longer be used. Transport Layer Security (TLS) is the current cryptographic protocol used to provide data integrity and security over networks by encrypting network connections at the transport layer. In many cases, people use the term SSL even when TLS is in fact the protocol being used.

the TCP/IP protocol is used, TLS is the preferred method of managing the security at the transport level.

Transport Layer Security (TLS) provides the most common means of interacting with a public key infrastructure (PKI) and certificates. This protocol provides for secure connections between the client and server and subsequent exchange of information. TLS can also provide server authentication (and optionally, client authentication) and confidentiality of information transfers. See Chapter 17 for a detailed explanation of the web-based security implementations.

The Internet Engineering Task Force (IETF) established the TLS working group in 1996 to develop a standard transport layer security protocol. The working group began with SSL version 3.0 as its basis and released RFC 2246, "The TLS Protocol Version 1.0," in 1999 as a proposed standard. The working group also published RFC 2712, "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)," as a proposed standard, and two RFCs on the use of TLS with HTTP. Like its predecessor SSL, TLS is a protocol that ensures privacy between communicating applications and their users on the Internet. When a server and client communicate, TLS ensures that no third party can eavesdrop or tamper with any message.

Tech Tip

Proper TLS Configuration

Protocols:

- *Deactivate SSLv2 and SSLv3, TLSv1.1.*
- *Activate TLSv1.2 and TLSv1.3.*

Configuration:

- *Deactivate client-initiated renegotiation (to prevent a specific form of DOS against TLS).*
- *Deactivate TLS compression, to prevent the attack known as CRIME (CVE-2012-4929).*
- *Activate support for forward secrecy.*

Cipher suites:

- *Deactivate cipher suites with keys that are shorter than 128 bit.*
- *Deactivate cipher suite RC4, to prevent attacks against RC4 (CVE-2013-2566).*

Certificates:

- *Certificates must include hostnames and domains in the CN and SAN fields.*
- *Use hash-algorithm SHA-256 instead of MD5 and SHA-1 for certificates.*

TLS is composed of two parts: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol provides connection security by using supported encryption methods. The TLS Record Protocol can also be used without encryption. The TLS Handshake Protocol allows the server and client to authenticate each other and to negotiate a session encryption algorithm and cryptographic keys before data is exchanged.

Though TLS is based on SSL and is sometimes referred to as SSL, they are not interoperable. However, the TLS protocol does contain a mechanism that allows a TLS implementation to back down to SSL 3.0. The difference

between the two is the way they perform key expansion and message authentication computations. The TLS Record Protocol is a layered protocol. At each layer, messages may include fields for length, description, and content. The Record Protocol takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a message authentication code (HMAC) to the data, encrypts it, and transmits the result. Received data is decrypted, verified, decompressed, and reassembled and then delivered to higher-level clients.

The TLS Handshake Protocol involves the following steps, which are summarized in Figure 6.2:

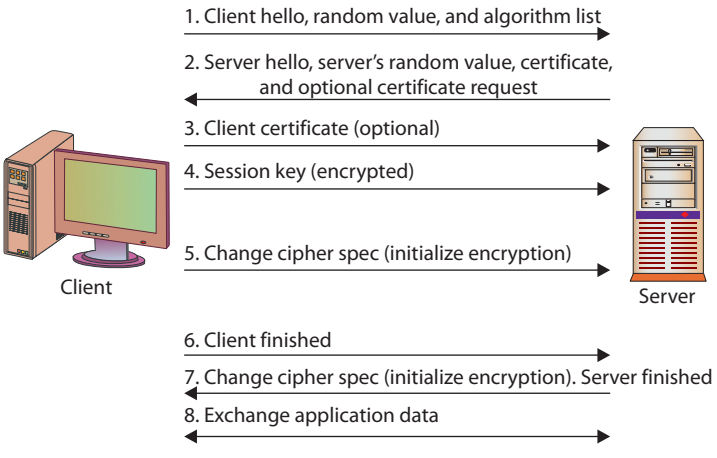


Figure 6.2 TLS Handshake Protocol

1. Client sends hello message, containing supported TLS options.
2. Server responds with a hello message selecting the TLS options. Then server sends its certificate.
3. Client sends client certificate (optional).
4. Client generates master secret and sends encrypted session key.
5. Client initializes encryption.
6. Client sends finished message to verify that key exchange was successful.
7. Server initializes encryption and sends message that it is ready for data.
8. Encrypted session begins.

Though it has been designed to minimize this risk, TLS still has potential vulnerabilities to a man-in-the-middle attack. A highly skilled and well-placed attacker can force TLS to operate at lower security levels. Regardless, through the use of validated and trusted certificates, a secure cipher suite can be selected for the exchange of data.

Once established, a TLS session remains active as long as data is being exchanged. If sufficient inactive time has elapsed for the secure connection to time out, it can be reinitiated.

TLS v1.3

TLS version 1.3 is a complete update to the TLS standard and is designed to address performance and security issues. TLS removed a lot of primitives and features that have contributed to weaknesses in previous versions. The cipher suites supported under TLS v1.3 are limited to AEAD ciphers, primarily to ward off a series of attacks against block ciphers. TLS v1.3 has other security enhancements to specifically protect against weak configurations and common vulnerabilities exploited by the attacks employed in DROWN, Vaudenay, Lucky 13, POODLE, SLOTH, and CRIME.

On the performance front, TLS v1.3 has made improvements in the handshake process, through the combination of steps, going from seven steps

down to five. Additionally, TLS supports zero round-trip time resumption (0-RTT), a method by which if a client and server have previously negotiated a TLS v1.3 connection, a reconnection can be achieved with one fewer round trip. While the elimination of round trips may seem like a small gain, the number of times these events happen behind the scenes is a lot, and these improvements can be noticeable especially on devices suffering from increased latency on each round trip.

Data in Transit/Motion

Transport encryption is used to protect **data in transit**, or data that is in motion. When data is being transported across a network, it is at risk of interception. An examination of the OSI networking model shows a layer dedicated to transport, and this abstraction can be used to manage end-to-end cryptographic functions for a communication channel. When the TCP/IP protocol is being used, TLS is one specific method of managing the security at the transport level. SSL is another example of transport layer security. Managing a secure layer of communications is an essential element in many forms of computer operations.

Data at Rest

Protecting **data at rest** is the most prominent use of encryption and is typically referred to as data encryption. Whole disk encryption of laptop data to provide security in the event of device loss is an example of data-at-rest protection. The same concept applies to data being stored in the cloud, where encryption can protect against unauthorized reading.

Data in transit, data at rest, and *data in use* are terms commonly used to describe states of data in a computing system. Understanding how to differentiate these terms based on their similarities and differences when it comes to cryptography is a very testable item.

Data in Use/Processing

Data in use is the term used to describe data that is stored in a nonpersistent state of either RAM, CPU caches, or CPU registers. Data in use is of increasing concern to security professionals as attacks such as RAM-scraping malware are occurring. Data in use is still data that requires protection, and in modern secure systems, this data can be encrypted. New techniques, such as Intel's SGX, promise a future where sensitive data can be protected from all other processes on a system, even those with higher levels of authority, such as root.

Implementation vs. Algorithm Selection

When using cryptography for protection of data, you need to include several factors in the implementation plan. One of the first decisions is in algorithm selection. Not only should you avoid deprecated algorithms, but you also need to match the algorithm to the intended use.

Cryptographic Service Provider

A **cryptographic service provider (CSP)** is a software library that implements cryptographic functions. CSPs implement encoding and decoding functions, which computer application programs may use, for example, to

implement strong user authentication or for secure e-mail. In Microsoft Windows, the Microsoft CryptoAPI (CAPI) is a CSP for all processes that need specific cryptographic functions. This provides a standard implementation of a complex set of processes.

Crypto Modules

A cryptographic module is a hardware or software device or component that performs cryptographic operations securely within a physical or logical boundary. **Crypto modules** use a hardware, software, or hybrid cryptographic engine contained within the boundary as well as cryptographic keys that do not leave the boundary, maintaining a level of security. Maintaining all secrets within a specified protected boundary has been a foundational element of a secure cryptographic solution.

Common Use Cases

Cryptographic services are being employed in more and more systems, and there are many common use cases associated with them. Examples include implementations to support situations such as low power, low latency, and high resiliency, as well as supporting functions such as confidentiality, integrity, and nonrepudiation.

Low-Power Devices

Low-power devices such as mobile phones and portable electronics are commonplace and have need for cryptographic functions. Cryptographic functions tend to take significant computational power, and special cryptographic functions, such as elliptic curve cryptography (ECC), are well suited for low-power applications.

Low Latency

Some use cases involve low-latency operations. This makes specialized cryptographic functions needed to support operations that have extreme time constraints. Stream ciphers are examples of low-latency operations.

High Resiliency

High-resiliency systems are characterized by functions that have the ability to resume normal operational conditions after an external disruption. The use of cryptographic modules can support resiliency through a standardized implementation of cryptographic flexibility.

Supporting Confidentiality

Protecting data from unauthorized reading is the definition of confidentiality. Cryptography is the primary means of protecting data confidentiality, data at rest, data in transit, and data in use.

Supporting Integrity

Times arise where the integrity of data is needed, such as during transfers. Integrity can demonstrate that data has not been altered. Message authentication codes (MACs) supported by hash functions are an example of cryptographic services supporting integrity.

Supporting Obfuscation

There are times when information needs to be obfuscated (that is, protected from casual observation). In the case of a program, obfuscation can protect the code from observation by unauthorized parties.

Supporting Authentication

Authentication is a property that deals with the identity of a party—be it a user, a program, or piece of hardware. Cryptographic functions can be employed to demonstrate authentication, such as the validation that an entity has a specific private key that's associated with a presented public key, thus proving identity.

Supporting Nonrepudiation

Nonrepudiation is a property that deals with the ability to verify that a message has been sent and received so that the sender (or receiver) cannot refute sending (or receiving) the information. An example of this in action is seen with the private key holder relationship. It is assumed that the private key never leaves the possession of the private key holder. Should this occur, it is the responsibility of the holder to revoke the key. Thus, if the private key is used, as evidenced by the success of the public key, then it is assumed that the message was sent by the private key holder. Therefore, actions that are signed cannot be repudiated by the holder.

Resource vs. Security Constraints

Cryptographic functions require system resources. Using the proper cryptographic functions for a particular functionality is important for both performance and resource reasons. Determining the correct set of security and resource constraints is an essential beginning step when planning a cryptographic implementation.

The commonly used hash functions in HMAC are MD5, SHA-1, and SHA-256. Although MD5 has been deprecated because of collision attacks, when it's used in the HMAC function, the attack methodology is not present and the hash function still stands as useful.

HMAC

HMAC is an acronym for keyed-*hash message authentication code,* a special form of message authentication code. Message authentication codes are used to determine whether a message has changed during transmission. Using a hash function for message integrity is common practice for many communications. When you add a secret key and crypto function, then the MAC becomes an HMAC, and you also have the ability to determine authenticity in addition to integrity.

S/MIME

In early 1996, the Internet Mail Consortium (IMC) was formed as a technical trade association pursuing cooperative use and enhancement of Internet electronic mail and messaging. An early goal of the IMC was to bring together the Department of Defense (DoD), along with its vendor community, and the commercial industry in order to devise a standard security

protocol acceptable to both. The Secure/Multipurpose Internet Mail Extensions (S/MIME) message specification is an extension to the MIME standard that provides a way to send and receive signed and encrypted MIME data. RSA Security created the first version of the S/MIME standard, using the RSA encryption algorithm and the Public Key Cryptography Standards (PKCS) series. The second version dates from 1998 but had a number of serious restrictions, including the restriction to 40-bit Data Encryption Standard (DES). The current version of the IETF standard is dated July 2004 and requires the use of Advanced Encryption Standard (AES).

Cross Check

E-mail Encryption

Want to understand e-mail encryption? Flip ahead to Chapter 17, which is on the Web, e-mail, and instant messaging, for more details on e-mail encryption. Then answer these questions:

- Why is it important to encrypt e-mail?
- What impacts can malicious code have on a business?
- Why is instant messaging a higher risk than e-mail?

The changes in the S/MIME standard have been so frequent that the standard has become difficult to implement until v3. Far from having a stable standard for several years that product manufacturers could have time to gain experience with, there were many changes to the encryption algorithms being used. Just as importantly, and not immediately clear from the IETF documents, the standard places reliance on more than one other standard for it to function. Key among these is the format of a public key certificate as expressed in the X.509 standard.

IETF S/MIME History

The S/MIME v2 specifications outline a basic strategy for providing security services for e-mail but lack many security features required by the DoD for use by the military. Shortly after the decision was made to revise the S/MIME v2 specifications, the DoD, its vendor community, and commercial industry met to begin development of the enhanced specifications. These new specifications would be known as S/MIME v3. Participants agreed that backward compatibility between S/MIME v3 and v2 should be preserved; otherwise, S/MIME v3–compatible applications would not be able to work with older S/MIME v2–compatible applications.

A minimum set of cryptographic algorithms was mandated so that different implementations of the new S/MIME v3 set of specifications could be interoperable. This minimum set must be implemented in an application for it to be considered "S/MIME compliant." Applications can implement additional cryptographic algorithms to meet their customers' needs, but the minimum set must also be present in the applications for interoperability with other S/MIME applications. Thus, users are not forced to use S/MIME-specified algorithms; they can choose their own. However, if the

application is to be considered S/MIME compliant, the standard algorithms must also be present.

IETF S/MIME v3 Specifications

Building on the original work by the IMC-organized group, the IETF has worked hard to enhance the S/MIME v3 specifications. The ultimate goal is to have the S/MIME v3 specifications receive recognition as an Internet standard. The current IETF S/MIME v3 set of specifications includes the following:

- Cryptographic Message Syntax (CMS)
- S/MIME v3 message specification
- S/MIME v3 certificate-handling specification
- Enhanced security services (ESS) for S/MIME

The CMS defines a standard syntax for transmitting cryptographic information about the contents of a protected message. Originally based on the PKCS #7 version 1.5 specification, the CMS specification was enhanced by the IETF S/MIME working group to include optional security components. Just as S/MIME v3 provides backward compatibility with v2, CMS provides backward compatibility with PKCS #7, so applications will be interoperable even if the new components are not implemented in a specific application.

Integrity, authentication, and nonrepudiation security features are provided by using digital signatures using the SignedData syntax described by the CMS. CMS also describes what is known as the EnvelopedData syntax to provide confidentiality of the message's content through the use of encryption. The PKCS #7 specification supports key encryption algorithms such as RSA. Algorithm independence is promoted through the addition of several fields to the EnvelopedData syntax in CMS, which is the major difference between the PKCS #7 and CMS specifications. The goal was to be able to support specific algorithms such as Diffie-Hellman and the Key Exchange Algorithm (KEA), which is implemented on the Fortezza Crypto Card developed for the DoD. One final significant change to the original specifications is the ability to include X.509 Attribute Certificates in the SignedData and EnvelopedData syntaxes for CMS.

An interesting feature of CMS is the ability to nest security envelopes to provide a combination of security features. As an example, a CMS triple-encapsulated message can be created in which the original content and associated attributes are signed and encapsulated within the inner SignedData object. The inner SignedData object is in turn encrypted and encapsulated within an EnvelopedData object. The resulting EnvelopedData object is then also signed and finally encapsulated within a second SignedData object, the outer SignedData object. Usually the inner SignedData object is signed by the original user and the outer SignedData object is signed by another entity, such as a firewall or a mail list agent, thus providing an additional level of security.

This triple encapsulation is not required of every CMS object. All that is required is a single SignedData object created by the user to sign a message or an EnvelopedData object if the user desired to encrypt a message.

Tech Tip

S/MIME in a Nutshell

S/MIME provides two security services to e-mail: digital signatures and message encryption. Digital signatures verify sender identity, and encryption can keep contents private during transmission. These services can be used independently of each other and provide the foundational basis for message security.

PGP

OpenPGP is a widely used e-mail encryption standard. A nonproprietary protocol for encrypting e-mail using public key cryptography, it is based on PGP, as originally developed by Phil Zimmermann, and is defined by the OpenPGP working group of the IETF proposed standard RFC 4880.

Pretty Good Privacy (PGP) is a popular program that is used to encrypt and decrypt e-mail and files. It also provides the ability to digitally sign a message so the receiver can be certain of the sender's identity. Taken together, encrypting and signing a message allows the receiver to be assured of who sent the message and to know that it was not modified during transmission. Public-domain versions of PGP have been available for years, as have inexpensive commercial versions.

PGP was one of the most widely used programs and was frequently used by both individuals and businesses to ensure data and e-mail privacy. It was developed by Philip R. Zimmermann in 1991 and quickly became a de facto standard for e-mail security. The popularity of PGP lead to the OpenPGP Internet standard, RFC 4880, and open source solutions. GNU Privacy Guard (GPG) is a common alternative to PGP in use today. What PGP started is now done in numerous apps, both free and commercial, protecting communications on a wide range of platforms from mobile devices to PCs.

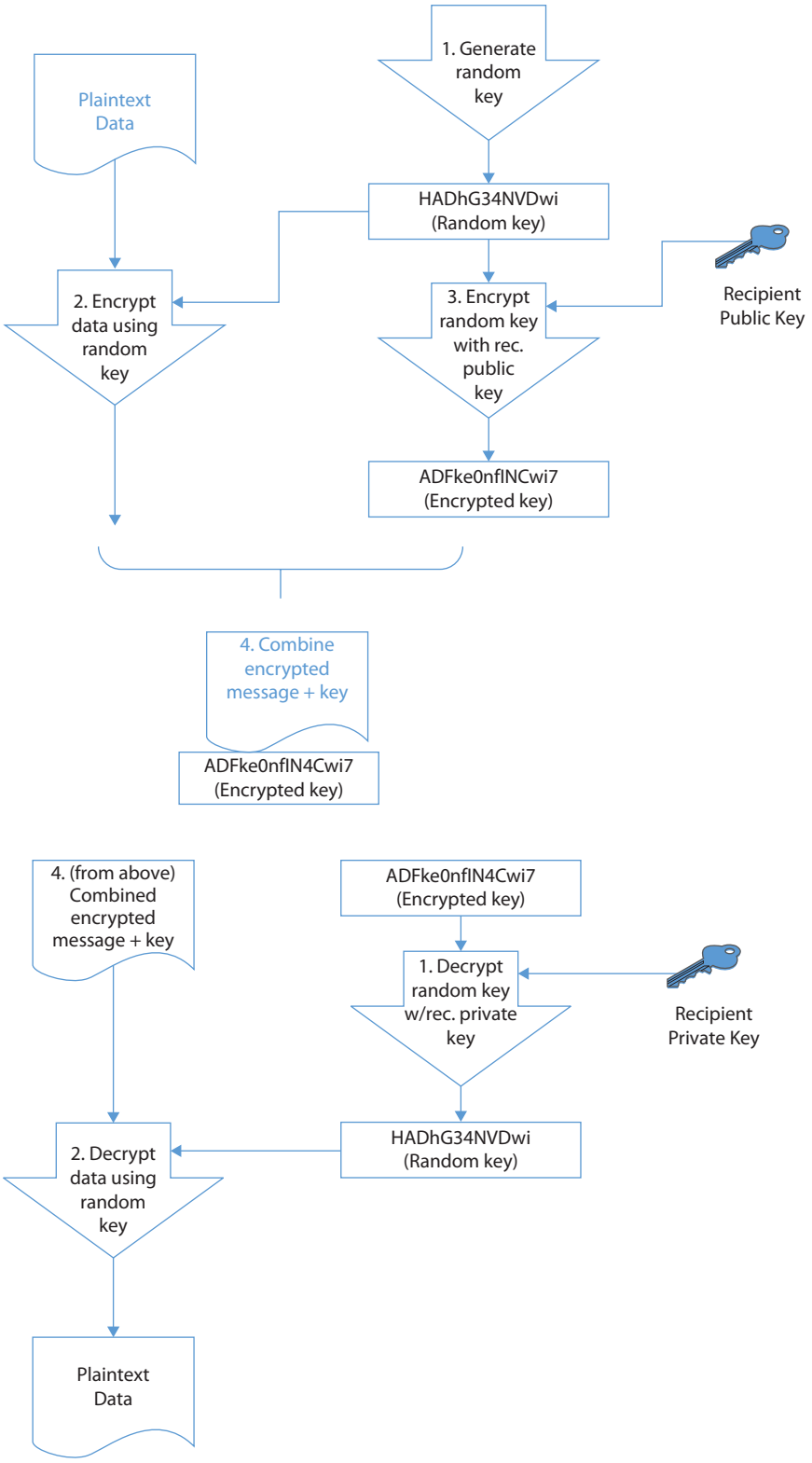
How PGP Works

PGP uses a variation of the standard public key encryption process. In public key encryption, an individual (here called the *creator*) uses the encryption program to create a pair of keys. One key is known as the *public key* and is designed to be given freely to others. The other key is called the *private key* and is designed to be known only by the creator. Individuals who want to send a private message to the creator encrypt the message using the creator's public key. The algorithm is designed such that only the private key can decrypt the message, so only the creator will be able to decrypt it.

This method, known as *public key* or *asymmetric encryption*, is time consuming. *Symmetric encryption* uses only a single key and is generally faster. It is because of this that PGP is designed the way it is. PGP uses a symmetric encryption algorithm to encrypt the message to be sent. It then encrypts the symmetric key used to encrypt this message with the public key of the intended recipient. Both the encrypted key and message are then sent. The receiver's version of PGP first decrypts the symmetric key with the private key supplied by the recipient and then uses the resulting decrypted key to decrypt the rest of the message.

PGP can use two different public key algorithms: Rivest-Shamir-Adleman (RSA) and Diffie-Hellman. The RSA version uses the International Data Encryption Algorithm (IDEA) and a short symmetric key to encrypt the message, and then uses RSA to encrypt the short IDEA key using the recipient's public key. The Diffie-Hellman version uses the Carlisle Adams and Stafford Tavares (CAST) algorithm to encrypt the message and the Diffie-Hellman algorithm to encrypt the CAST key. To decrypt the message, the reverse is performed. The recipient uses their private key to decrypt the IDEA or CAST key and then uses that decrypted key to decrypt the message. These are both illustrated in Figure 6.3.

To generate a digital signature, PGP takes advantage of another property of public key encryption schemes. Normally, the sender encrypts



• **Figure 6.3** How PGP works for encryption

using the receiver's public key and the message is decrypted at the other end using the receiver's private key. The process can be reversed so that the sender encrypts (signs) with their own private key. The receiver then decrypts the message with the sender's public key. Because the sender is the only individual who has a key that will correctly be decrypted with the sender's public key, the receiver knows that the message was created by the sender who claims to have sent it. The way PGP accomplishes this task is to generate a hash value from the user's name and other signature information. This hash value is then encrypted with the sender's private key known only by the sender. The receiver uses the sender's public key, which is available to everyone, to decrypt the hash value. If the decrypted hash value matches the hash value sent as the digital signature for the message, then the receiver is assured that the message was sent by the sender who claims to have sent it.

Typically, versions of PGP contain a user interface that works with common e-mail programs such as Microsoft Outlook. If you want others to be able to send you an encrypted message, you need to register your public key, generated by your PGP program, with a PGP public key server. Alternatively, you have to either send your public key to all those who want to send you an encrypted message or post your key to some location from which they can download it, such as your web page. Note that using a public key server is the better method, for all the reasons of trust described in the discussion of PKIs in Chapter 7.

Tech Tip

Where Can You Use PGP?

For many years the U.S. government waged a fight over the exportation of PGP technology, and for many years its exportation was illegal. Today, however, PGP-encrypted e-mail can be exchanged with most users outside the United States, and many versions of PGP are available from numerous international sites. Of course, being able to exchange PGP-encrypted e-mail requires that the individuals on both sides of the communication have valid versions of PGP. Interestingly, international versions of PGP are just as secure as domestic versions—a feature that is not true of other encryption products. It should be noted that the freeware versions of PGP are not licensed for commercial purposes.

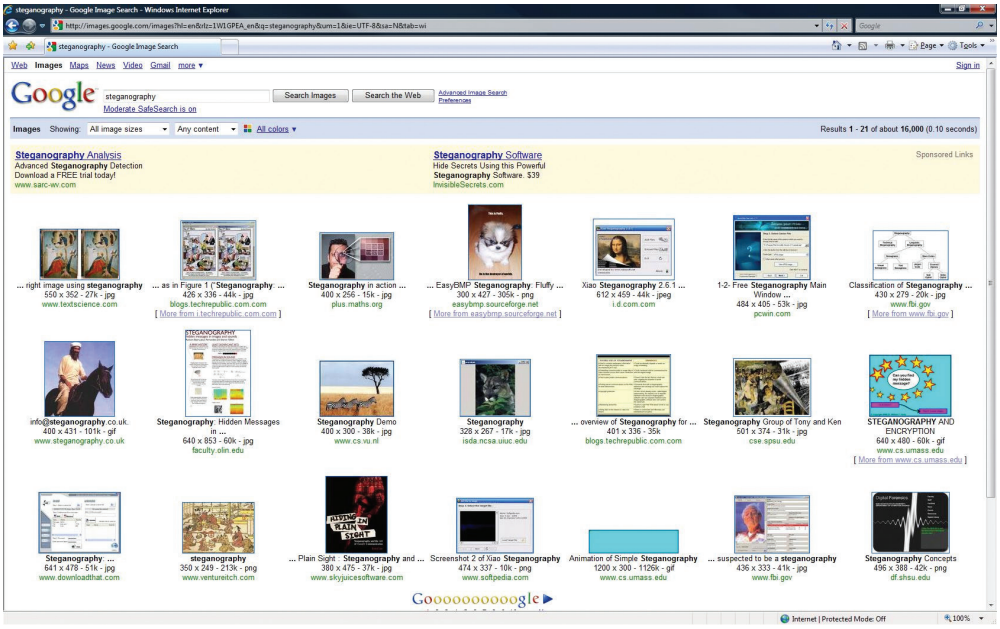
Steganography

Steganography is a technique used to hide secret data within non-secret data in order to avoid detection.

Steganography, an offshoot of cryptography technology, gets its meaning from the Greek word *steganos*, meaning covered. Invisible ink placed on a document hidden by innocuous text is an example of a steganographic message. Another example is a tattoo placed on the top of a person's head, visible only when the person's hair is shaved off.

Hidden writing in the computer age relies on a program to hide data inside other data. The most common application is the concealing of a text message in a picture file. The Internet contains multiple billions of image files, allowing a hidden message to be located almost anywhere without being discovered. Because not all detection programs can detect every kind

of steganography, trying to find the message in an Internet image is akin to attempting to find a needle in a haystack the size of the Pacific Ocean; even a Google search for steganography returns thousands of images:



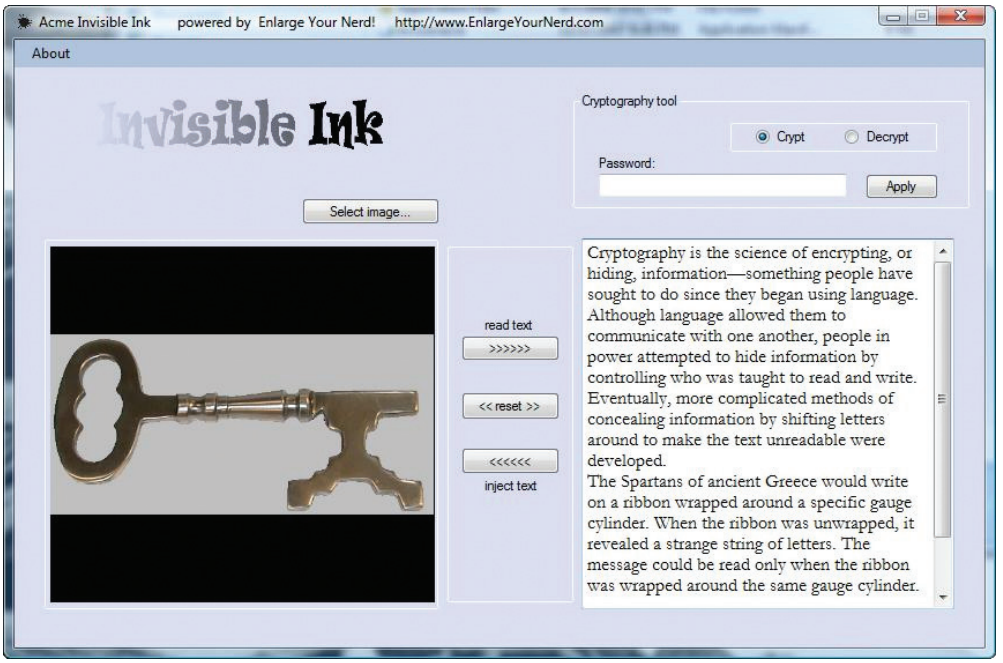
The nature of the image files also makes a hidden message difficult to detect. Although it is most common to hide messages inside images, they can also be hidden in video and audio files.

The advantage to steganography over the use of encryption alone is that the messages do not attract attention, and this difficulty in detecting the hidden message provides an additional barrier to analysis. The data that is hidden in a steganographic message is frequently also encrypted so that if it is discovered, the message will remain secure. Steganography has many uses, but the most publicized uses are to hide illegal material, often pornography, or allegedly for covert communication by terrorist networks.

Steganographic encoding can be used in many ways and through many different media. Covering them all is beyond the scope for this book, but we will discuss one of the most common ways to encode into an image file: LSB encoding. LSB, or *least significant bit*, is a method of encoding information into an image while altering the actual visual image as little as possible. A computer image is made up of thousands or millions of pixels, all defined by 1's and 0's. If an image is composed of Red Green Blue (RGB) values, each pixel has an RGB value represented numerically from 0 to 255. For example, 0,0,0 is black, and 255,255,255 is white, which can also be represented as 00000000, 00000000, 00000000 for black and 11111111, 11111111, 11111111 for white. Given a white pixel, editing the least significant bit of the pixel to 11111110, 11111110, 11111110 changes the color. The change in color is undetectable to the human eye, but in an image with a million pixels, this creates a 125KB area in which to store a message.

Some popular steganography detection tools include Stegdetect, StegSecret, StegSpy, and the family of SARC (Steganography Analysis and Research Center) tools. All of these tools use detection techniques based on the same principle: pattern detection. By looking for known steganographic

encoding schemes or artifacts, they can potentially detect embedded data. Additionally, steganography insertion tools can be used to attempt to decode images with suspected hidden messages. Invisible Ink is a small program for steganographic insertion of messages and then the extraction of those messages, as illustrated here:



Secure Protocols

Protocols act as a common language, allowing different components to talk using a common, known set of commands. Many different protocols exist, all of which are used to achieve specific communication goals.

DNSSEC

The Domain Name Service (DNS) is a protocol for the translation of names into IP addresses. When users enter a name such as www.example.com, the DNS system converts this name into the actual numerical IP address. DNS records are also used for e-mail delivery. The DNS protocol uses UDP over port 53 for standard queries, although TCP can be used for large transfers such as zone transfers. DNS is a hierarchical system of servers, from local copies of records, up through Internet providers to root-level servers. DNS is one of the primary underlying protocols used on the Internet and is involved in almost all addressing lookups. The problem with DNS is that requests and replies are sent in plaintext and are subject to spoofing.

DNSSEC is a set of extensions to the DNS protocol, which through the use of cryptography enables origin authentication of DNS data, authenticated denial of existence, and data integrity, but not does not extend to availability or confidentiality. DNSSEC records are signed so that all DNSSEC responses are authenticated but not encrypted. This prevents unauthorized DNS responses from being interpreted as correct.

Tech Tip

Common Ports

SSH	*TCP 22*
LDAPS	*TCP 636 or 3269*
FTPS	*TCP 989 and 990*
SFTP	*TCP 22 (SSH)*
SNMP	*UDP 161 and 162 or TCP 161 and 162*
HTTPS	*TCP 443*
Secure POP	*TCP 995*
Secure IMAP	*TCP 993*
Secure SMTP	*TCP 465*

SSH

The **Secure Shell (SSH)** protocol is an encrypted remote terminal connection program used for remote connections to a server. SSH uses asymmetric encryption but generally requires an independent source of trust with a server, such as manually receiving a server key, to operate. SSH uses TCP port 22 as its default port.

S/MIME

As previously mentioned, **S/MIME (Secure/Multipurpose Internet Mail Extensions)** is a standard for public key encryption and signing of Multipurpose Internet Mail Extensions data in e-mails. S/MIME is designed to provide cryptographic protections to e-mails and is built into the majority of modern e-mail software to facilitate interoperability.

SRTP

The **Secure Real-time Transport Protocol (SRTP)** is a network protocol for securely delivering audio and video over IP networks. SRTP uses cryptography to provide encryption, message authentication, and integrity, as well as replay protection to the RTP data.

LDAPS

By default, LDAP traffic is transmitted insecurely. You can make LDAP traffic secure by using it with SSL/TLS. Commonly, LDAP is enabled over SSL (LDAPS) by using a certificate from a trusted certificate authority (CA). **Lightweight Directory Access Protocol Secure (LDAPS)** involves the use of an SSL tunnel to connect LDAP services. Technically, this method was retired with LDAPv2 and replaced with Simple Authentication and Security Layer (SASL) in LDAPv3. SASL is a standard method of using TLS to secure services across the Internet. LDAP is the primary protocol for transmitting directory information. Directory services may provide any organized set of records, often with a hierarchical structure, and are used in a wide variety of situations, including Active Directory data sets. LDAPS communication occurs over TCP port 636. LDAPS communication to a global catalog server occurs over TCP 3269. When connecting to port 636 or 3269, SSL/TLS is negotiated before any LDAP traffic is exchanged.

FTPS

FTPS is the implementation of FTP over an SSL/TLS secured channel. This supports complete FTP compatibility, yet provides the encryption protections enabled by SSL/TLS. FTPS uses TCP ports 989 and 990.

SFTP

SFTP involves the use of FTP over an SSH channel. This leverages the encryption protections of SSH to secure FTP transfers. Because of its reliance on SSH, it uses TCP port 22.

SNMPv3

The **Simple Network Management Protocol version 3 (SNMPv3)** is a standard for managing devices on IP-based networks. SNMPv3 was developed to specifically address the security concerns and vulnerabilities of SNMPv1 and SNMPv2. SNMP is an application layer protocol, part of the IP suite of protocols, and can be used to manage and monitor devices, including network devices, computers, and other devices connected to the IP network. All versions of SNMP require ports 161 and 162 to be open on a firewall. Most SNMP systems use UDP, but TCP can be used as well.

TLS

As previously mentioned, **Transport Layer Security (TLS)** is an IETF standard for the employment of encryption technology and replaces SSL. Using the same basic principles, TLS updates the mechanisms employed by SSL. Although sometimes referred to as SSL, TLS is a separate standard. The standard port for TLS is undefined because it depends on what the protocol being protected uses—for example, HTTP (port 80) becomes port 443 when it is HTTPS. The current versions of TLS are 1.2 and 1.3.

HTTPS

Most web activity occurs using the Hypertext Transfer Protocol (HTTP), but this protocol is prone to interception. **HTTP Secure (HTTPS)** uses either SSL or TLS to secure the communication channel. Originally developed by Netscape Communications and implemented in its browser, HTTPS has since been incorporated into most common browsers. HTTPS uses the standard TCP port 443 for TCP/IP communications rather than the standard port 80 used for HTTP. As previously discussed, because of vulnerabilities in SSL, only TLS is recommended for HTTPS today.

Secure POP/IMAP

Secure POP3 and **Secure IMAP** are basically POP3 and IMAP, respectively, over an SSL/TLS session. Secure POP3 utilizes TCP port 995 and Secure IMAP uses TCP port 993. Encrypted data from the e-mail client is sent to the e-mail server over an SSL/TLS session. With the deprecation of SSL, TLS is the preferred protocol today. If e-mail connections are started in nonsecure mode, the STARTTLS directive tells the clients to change to the secure ports.

IMAP uses port 143, but SSL/TLS-encrypted IMAP uses port 993. POP uses port 110, but SSL/TLS-encrypted POP uses port 995. SMTP uses port 25, but SSL/TLS-encrypted SMTP uses port 465.

IPSec

IP Security (IPSec) is a collection of IP security features designed to introduce security at the network or packet-processing layer in network communication. Other approaches have attempted to incorporate security at higher levels of the TCP/IP suite, such as at the level where applications reside. IPSec is designed to provide secure IP communications over the Internet. In essence, IPSec provides a secure version of the IP by introducing authentication and encryption to protect Layer 4 protocols. IPSec is optional for

IPv4 but is required for IPv6. Obviously, both ends of the communication need to use IPSec for the encryption/decryption process to occur.

IPSec provides two types of security service to ensure authentication and confidentiality for either the data alone (referred to as IPSec *transport mode*) or for both the data and header (referred to as *tunnel mode*). See Chapter 11 for more detail on tunneling and IPSec operation. IPSec introduces several new protocols, including the Authentication Header (AH), which basically provides authentication of the sender, and the Encapsulating Security Payload (ESP), which adds encryption of the data to ensure confidentiality. IPSec also provides for payload compression before encryption using the IP Payload Compression Protocol (IPcomp). Frequently, encryption negatively impacts the ability of compression algorithms to fully compress data for transmission. By providing the ability to compress the data before encryption, IPSec addresses this issue.

IPSec is a secure network protocol suite that is used to authenticate and encrypt data packets sent between systems. It provides authentication, integrity, and confidentiality.

Secure Protocol Use Cases

Protocols are used to allow parties to have a common understanding of how communications will be handled, and they define the expectations for each party. Because different use cases have different communication needs, different protocols will be used by different use cases. Work has been afoot to standardize some general-purpose security protocols, ones that can be reused over and over instead of new ones being invented for each use case. The Simple Authentication and Security Layer (SASL) effort is an example of that—it's a standardized method of invoking a TLS tunnel to secure a communication channel. This method is shown to work with a wide range of services—currently more than 15, and increasing.

The remainder of this section examines some common secure protocol use cases and the associated secure protocols used in them. Although the protocols have been discussed earlier, these are the use cases.

Voice and Video

Voice and video are frequently streaming media and, as such, have their own protocols for the encoding of data streams. The *Secure Real-time Transport Protocol (SRTP)* can be used for securely delivering audio and video over IP networks. SRTP is covered in RFC 3711 (https://tools.ietf.org/html/rfc3711).

Time Synchronization

NTP (Network Time Protocol) is the standard for time synchronization across servers and clients. NTP has no assurance against a man-in-the-middle attack, and although this has raised concerns over the implications, to date, nothing has been done to secure NTP directly or to engineer an out-of-band security check. If an organization is hypersensitive to this risk, it could enclose all time communications using a TLS tunnel, although this is not an industry practice.

E-mail and Web

E-mail and the Web are both native plaintext-based systems. With the need for secure web connections, SSL and TLS are available, as denoted by the HTTPS protocol. The use of SSL/TLS is widespread and common. Also, it is worth remembering that SSL is no longer considered secure. E-mail is a bit more complicated to secure, and the best option is to use S/MIME, as discussed previously in this chapter.

File Transfer

Secure file transfer can be accomplished via a wide range of methods, ensuring the confidentiality and integrity of file transfers across networks. FTP is not secure, but as previously discussed, secure alternatives such as SFTP and FTPS exist and can be used.

Directory Services

Directory services use LDAP as the primary protocol. When security is required, LDAPS is a common option, as described previously. Directory services are frequently found behind the scenes with respect to logon information.

Remote Access

Remote access is the means by which users can access computer resources across a network. Securing remote access can be done via many means; some for securing the authentication process, and others for the actual data access itself. As with many situations requiring securing communication channels, or data in transit, SSL/TLS is commonly employed.

Domain Name Resolution

Domain name resolution is performed primarily by the DNS protocol. DNS is a plaintext protocol, and the secure version, DNSSEC, is not widely deployed globally as yet. For local deployments, DNSSEC has been available in Windows Active Directory domains since 2012.

Routing and Switching

Routing and switching are the backbone functions of networking in a system. Managing the data associated with networking is the province of SNMPv3. SNMPv3 enables applications to manage data associated with networking and devices.

Network Address Allocation

Managing network address allocation functions in a network requires multiple decision criteria, including the reduction of complexity and the management of device names and locations. SNMPv3 has many functions that

can be employed to manage the data flows of this information to management applications that can assist administrators in network assignments.

IP addresses can be allocated either statically (with a manual, fixed IP to each device solution) or via Dynamic Host Configuration Protocol (DHCP, which allows the automation of assigning IP addresses). In some cases, a mix of static and DHCP is used. IP address allocation is part of proper network design, which is crucial to the performance and expandability of a network. Learn how to properly allocate IP addresses for a new network—and learn your options if you run out of IP addresses.

Subscription Services

Subscription services involve the management of data flows to and from a system based on either a push (publish) or pull (subscribe) model. Managing what data elements are needed by which nodes is a problem that can be tackled using directory services such as LDAP. Another use of subscription services is the Software as a Service (SaaS) model, where software is licensed on a subscription basis. The actual software is hosted centrally (commonly in the cloud), and user access is based on subscriptions. This is becoming a common software business model.

Cryptographic Attacks

Attacks against the cryptographic system are referred to as *cryptographic attacks*. These attacks are designed to take advantage of two specific weaknesses. The first is on the side of a user. Users widely view cryptography as magic, or otherwise incomprehensible stuff, leading them to trust without valid reasons. The second factor is aimed at algorithmic weaknesses that can be exploited. Although understood by computer scientists, they are frequently overlooked by developers. A variant and much more likely algorithmic weakness is in the actual implementation of the algorithm in code. Errors in coding encryption algorithms can result in systems that appear to work, but in reality are weak or incorrect implementations. As we will explore in Chapter 19, during software development it is important to use vetted libraries for cryptographic functions and proven algorithms.

There is a wide array of known cryptographic attack mechanisms. From known-plaintext attacks to chosen-plaintext attacks, ciphertext-only attacks, chosen-ciphertext attacks (both CCA1 and CCA2), chosen-key attacks, and meet-in-the-middle attacks, the list of things a crypto-developer needs to be aware of and protect against can be intimidating. This is one of the reasons one should only use approved crypto libraries, where implementations are vetted and have specific protections against common modes of attack. There are implementation-based attack methods, such as man-in-the-middle attacks, the KRACK attack against WPA2, replay attacks, and side-channel attacks, that also require consideration and protection. At the end of the day, the very complexity that enables the functions of modern cryptography can obscure and enable these attack vectors.

Birthday

The **birthday attack** is a special type of brute force attack that gets its name from something known as the birthday paradox, which states that in a group of at least 23 people, the chance that two individuals will have the same birthday is greater than 50 percent. Mathematically, we can use the equation $1.25k^{1/2}$ (with k equaling the size of the set of possible values), and in the birthday paradox, k would be equal to 365 (the number of possible birthdays). This same phenomenon applies to passwords, with k (number of passwords) being quite a bit larger. This is the result of having many combinations of two items from a large group, making the number of potential matching states higher. The key to the birthday attack is not to search for a match to a specific item, but rather to find any two items with the same key.

Known Plaintext/Ciphertext

If an attacker has the original plaintext and ciphertext for a message, they can determine the key used through brute force attempts through the keyspace. *Known plaintext/ciphertext* attacks can be difficult to mitigate, as some messages are particularly prone to this problem. In the event of having known messages, such as the German weather reports, it was possible using cryptanalysis techniques to eventually determine the Enigma machine rotor combinations, leading to the breakdown of that system (see Tech Tip "Weak Keys"). Modern cryptographic algorithms have protections included in the implementations to guard against this form of attack. One is the use of large keyspaces, making the brute force spanning of the keyspace, or even a significant portion of it, no longer possible.

Chosen Cipher Text Attack

A common attack methodology used against encryption schemes is the chosen cipher text attack. In this attack, an attacker presents a chosen cipher text to the system and, using the information from a series of these decryption efforts, can perform cryptanalysis to attempt to determine the key. If blocks are padded as part of the process, then by playing with the padding, information can be recovered. Although these attacks are highly technical and can require significant resources to interpret, the scale of web-based systems can result in millions of these messages being sent and received, providing significant information for an attacker to use in key discovery. One of the primary drivers of AEAD was its resistance to this type of attack.

Weak Implementations

Weak implementations are another problem associated with backward compatibility. The best example of this is SSL. SSL, in all of its versions, has now fallen to attackers. TLS, an equivalent methodology that does not suffer from these weaknesses, is the obvious solution; yet many sites still employ SSL. Cryptography has long been described as an arms race between attackers and defenders, with multiple versions and improvements over the years. Whenever an older version is allowed, there is a risk associated with weaker implementations.

Tech Tip

Weak Keys

Keys are instrumental in the functioning of the algorithm, and there are cases where a key can be considered a weak key because it forces the algorithm output to an undesired path, resulting in an encrypted version that is easily broken. Weak rotor combinations led to weak encryption with Enigma machines in WWII. Another example is in Triple DES, where certain key patterns can lead to weak results. For instance, the DES key is broken into 16 subkeys in use. If the key creates 16 identical subkeys, this will result in weaker-than-normal results. Another example is in Blowfish, where S-boxes are key dependent and certain keys can create weak S-boxes. The verification of proper keys for an algorithm needs to be one of the functions of the key-generation mechanism; otherwise, real-life operation may result in failure.

Tech Tip

Chosen Plain Text Attack

If you can get your adversary to send a known message (chosen plaintext), then this can assist in breaking the key. This was done in several cases in WWII both against the Germans to break Enigma keys (using weather reports) and against Japan (using a fake set of intelligence about broken equipment).

Meet-in-the-Middle Attacks

The **meet-in-the-middle attack** involves attacking the problem from two directions and looking for the match. The plaintext is encrypted with every possible key in one direction. The cryptographic message is decrypted with every possible key in the other direction. The result of the comparison can help to discover which algorithm is used and the secret key that was used.

Replay

Replay attacks work against cryptographic systems like they do against other systems. If one can record a series of packets and then replay them, what was valid before may well be valid again. There are a wide range of defenses against replay attacks, and therefore this should not be an issue. However, developers who do not follow best practices can create implementations that lack replay protections, enabling this attack path to persist.

Downgrade

As part of a TLS/SSL setup, there is a specification of the cipher suite to be employed. This is done to enable the highest form of encryption that both the server and browser can support. In a **downgrade attack**, the attacker takes advantage of a commonly employed principle to support backward compatibility, to downgrade the security to a lower or nonexistent state.

Tech Tip

Password Defenses

There are two players in password defenses: the system designer and the user.

A system designer needs to ensure that a password system does the proper handling, storage (of the salted hash only) of passwords, and logging of activity. The designer also needs to invoke lockouts if incorrect passwords are entered. Failure to lock out incorrect choices allows unlimited attempts on the part of an attacker.

Users need to be trained in good password hygiene: how to create a good password, not to share passwords between accounts, and to watch for phishing attempts that try to obtain their password.

Passwords are still useful for security, but they need to be instantiated and used properly.

Collision

A **collision attack** is where two different inputs yield the same output of a hash function. Through making "invisible to the user" changes to a digital file and creating many copies, then using the birthday attack to find a collision between any two of the many versions, one has a chance to create a file with changed visible content but identical hashes.

Password Attacks

The most common form of authentication is the user ID and password combination. Although it is not inherently a poor mechanism for authentication, the combination can be attacked in several ways. All too often, these attacks yield favorable results for the attacker—not as a result of a weakness in the scheme, but usually due to the user not following good password procedures.

Poor Password Choices

The least technical of the various password-attack techniques consists of the attacker simply attempting to guess the password of an authorized user of the system or network. It is surprising how often this simple method works, and the reason it does is because people are notorious for picking poor passwords. Users need to select a password that they can remember, so they create simple passwords, such as their birthday, their mother's maiden name, the name of their spouse or one of their children, or even

simply their user ID itself. All it takes is for the attacker to obtain a valid user ID (often a simple matter, because organizations tend to use an individual's names in some combination—first letter of their first name combined with their last name, for example) and a little bit of information about the user before guessing can begin. Organizations sometimes make it even easier for attackers to obtain this sort of information by posting the names of their "management team" and other individuals, sometimes with short biographies, on their websites.

Even if a person doesn't use some personal detail as their password, the attacker may still get lucky because many people use a common word for their password. Attackers can obtain lists of common passwords—a number of such lists exist on the Internet. Words such as "password" and "secret" have often been used as passwords. Names of favorite sports teams also often find their way onto lists of commonly used passwords.

Tech Tip

Rainbow Tables

Rainbow tables *are precomputed hash tables that enable looking up small text entries via their hash values. This makes hashed passwords "reversible." The use of rainbow tables works for small passwords (fewer than 10 characters) and is very fast. Salting passwords is one of the defenses against these tables.*

Rainbow Tables

Rainbow tables are precomputed tables or hash values associated with passwords. This can change the search for a password from a computational problem to a lookup problem. This can tremendously reduce the level of work needed to crack a given password. The best defense against rainbow tables is the use of salted hashes, as the addition of a salt value increases the complexity of the problem by making the precomputing process not replicable between systems.

Dictionary

Another method of determining passwords is to use a password-cracking program that uses a list of dictionary words to try to guess the password, hence the name *dictionary attack*. The words can be used by themselves, or two or more smaller words can be combined to form a single possible password. A number of commercial and public domain password-cracking programs employ a variety of methods to crack passwords, including using variations on the user ID.

These programs often permit the attacker to create various rules that tell the program how to combine words to form new possible passwords. Users commonly substitute certain numbers for specific letters. If the user wanted to use the word *secret* for a password, for example, the letter *e* could be replaced with the number *3*, yielding *s3cr3t*. This password will not be found in the dictionary, so a pure dictionary attack would not crack it, but the password is still easy for the user to remember. If a rule were created that tried all words in the dictionary and then tried the same words substituting the number *3* for the letter *e*, however, the password would be cracked.

Rules can also be defined so that the cracking program will substitute special characters for other characters or combine words. The ability of the attacker to crack passwords is directly related to the method the user employs to create the password in the first place, as well as the dictionary and rules used.

Brute Force

If the user has selected a password that is not found in a dictionary, even if various numbers or special characters are substituted for letters, the only

way the password can be cracked is for an attacker to attempt a *brute force attack,* in which the password-cracking program attempts all possible password combinations.

The length of the password and the size of the set of possible characters in the password will greatly affect the time a brute force attack will take. A few years ago, this method of attack was very time consuming, since it took considerable time to generate all possible combinations. With the increase in computer speed, however, generating password combinations is much faster, making it more feasible to launch brute force attacks against certain computer systems and networks.

A brute force attack on a password can take place at two levels: it can attack a system where the attacker is attempting to guess the password at a login prompt, or it can attack against the list of password hashes contained in a password file. The first attack can be made more difficult if the account locks after a few failed login attempts. The second attack can be thwarted if the password file is securely maintained so that others cannot obtain a copy of it.

Online vs. Offline

When the brute force attack occurs in real time against a system, it is frequently being done to attack a single account with multiple examples of passwords. Success or failure is determined by the system under attack: either you get in or you don't. Online brute force attacks tend to be very noisy and easy to see by network security monitoring. They are also limited by system response time and bandwidth.

Offline, brute force can be employed to perform hash comparisons against a stolen password file. This has the challenge of stealing the password file, but if accomplished, it is possible to use high-performance, GPU-based, parallel machines to try passwords at very high rates and against multiple accounts at the same time.

Hybrid Attack

A *hybrid* password attack is a system that combines the preceding methods. Most cracking tools have this option built in, first attempting a dictionary attack and then moving to brute force methods.

Password Spraying Attacks

Password *spraying* is an attack that uses a limited number of commonly used passwords and applies them to a large number of accounts. Traditional brute force attacks attempt to gain unauthorized access to a single account by guessing the password. Spraying is the reverse of this, using a limited number of passwords and trying them against all the accounts. This is a useful attack when you don't care which account you get, and it's fairly successful when given a large set of accounts. Defending against this attack is important in organizations because if one account is breached, it is the foothold that is needed to gain entry.

Plaintext/Unencrypted

Passwords that are stored are subject to retrieval. Any time a system can send you a copy of your password, there is a security issue. Plaintext

password attacks are those taken against these specific issues. Lest anyone think that this is only a problem from rogue systems or programs, even mainstream systems can fall prey to this trap. Microsoft allows administrators to push out passwords for local accounts via group policy preferences. To protect the passwords, they are encrypted using Advanced Encryption Standard (AES). For reasons of compatibility with other systems, Microsoft published the AES key. See the problem?

In Microsoft Windows systems, Mimikatz is a security tool that can extract Kerberos tickets from memory, and it also possesses the ability to extract plaintext passwords from process dumps of the LSASS process. This means that by using the security tools ProcDump and Mimikatz, one can harvest plaintext passwords from a system.

Other Standards

Many additional standards are associated with information security that are not specifically or solely associated with PKI and/or cryptography. The remainder of the chapter introduces these standards and protocols.

FIPS

The **Federal Information Processing Standards Publications (FIPS PUBS, or simply FIPS)** describe various standards for data communication issues. These documents are issued by the U.S. government through the National Institute of Standards and Technology (NIST), which is tasked with their development. NIST creates these publications when a compelling government need requires a standard for use in areas such as security or system interoperability and no recognized industry standard exists. Three categories of FIPS PUBS are currently maintained by NIST:

- Hardware and software standards/guidelines
- Data standards/guidelines
- Computer security standards/guidelines

These documents require that products sold to the U.S. government comply with one (or more) of the FIPS standards. The standards can be obtained from www.nist.gov/itl/fips.cfm.

FIPS 140-2 relates to specific cryptographic standards for the validation of components used in U.S. government systems. Systems can be accredited to the FIPS 140-2 standard to demonstrate levels of security from "approved algorithms" to higher levels that include additional protections up to and including physical security and tamperproof mechanisms.

Common Criteria

The Common Criteria (CC) for Information Technology Security is the result of an effort to develop a joint set of security processes and standards that can be used by the international community. The major contributors to the CC are the governments of the United States, Canada, France, Germany, the Netherlands, and the United Kingdom. The CC also provides a listing of laboratories that apply the criteria in testing security products. Products that are evaluated by one of the approved laboratories receive an Evaluation Assurance Level of EAL1 through EAL7 (EAL7 is the highest level), with EAL4, for example, designed for environments requiring a moderate

to high level of independently assured security, and EAL1 being designed for environments in which some confidence in the correct operation of the system is required but where the threats to the system are not considered serious. The CC also provides a listing of products by function that have performed at a specific EAL.

ISO/IEC 27002 (Formerly ISO 17799)

ISO/IEC 27002 is a very popular and detailed standard for creating and implementing security policies. ISO/IEC 27002 was formerly ISO 17799, which was based on version 2 of the British Standard 7799 (BS7799), published in May 1999. With the increased emphasis placed on security in both the government and industry in recent years, many organizations are now training their audit personnel to evaluate their organizations against the ISO/IEC 27002 standard. The standard is divided into 12 sections, each containing more detailed statements describing what is involved for that topic:

- **Risk assessment** Determine the impact of risks
- **Security policy** Guidance and policy provided by management
- **Organization of information security** Governance structure to implement security policy
- **Asset management** Inventory and classification of assets
- **Human resources security** Policies and procedures addressing security for employees, including hires, changes, and departures
- **Physical and environmental security** Protection of the computer facilities
- **Communications and operations management** Management of technical security controls in systems and networks
- **Access control** Restriction of access rights to networks, systems, applications, functions, and data
- **Information systems acquisition, development, and maintenance** Building security into applications
- **Information security incident management** Anticipating and responding appropriately to information security breaches
- **Business continuity management** Protecting, maintaining, and recovering business-critical processes and systems
- **Compliance** Ensuring conformance with information security policies, standards, laws, and regulations

Chapter 6 Review

Chapter Summary

After reading this chapter and completing the exercises, you should understand the following about applied cryptography.

Learn the elements involved in the correct use of cryptography

- Cryptography is used to ensure confidentiality, integrity, authentication, and nonrepudiation.
- How common cryptographic applications, including digital signatures, digital rights management, and file and database encryption systems, are used to secure information in systems and during communications.

Examine cipher suites and common uses

- Understand algorithm selection and use to implement secret communications and key exchanges.
- The common use cases of low-power devices, low latency, high resiliency, and supporting confidentiality, integrity, obfuscation, and authentication.
- Encryption for transport, including data in transit, data at rest, and data in use.

Learn cryptographic attack methods

- Common attack methods, including birthday, collision, downgrade, known-plaintext, and meet-in-the-middle attacks.
- An examination of password attacks, including poor password choices, rainbow tables, dictionary, brute force, and hybrid attacks.
- Digital rights management (DRM) uses some form of asymmetric encryption that allows an application to determine if you are an authorized user of the digital content you are trying to access. For example, DVDs and certain digital music formats such as AACS use DRM.
- Cipher suites provide information to assist developers in choosing the correct methods to achieve desired levels of protection.

Key Terms

Bcrypt *(157)*
birthday attack *(175)*
cipher suite *(153)*
collison attack *(176)*
cryptographic service provider (CSP) *(160)*
crypto modules *(161)*
data at rest *(160)*
data in transit *(160)*
data in use *(160)*
digital rights management (DRM) *(151)*
digital signature *(150)*
DNSSEC *(169)*
downgrade attack *(176)*
ephemeral keys *(157)*
Federal Information Processing Standards Publications (FIPS PUBS or simply FIPS) *(179)*
FTPS *(170)*
HMAC-based one-time password (HOTP) *(150)*
Hypertext Transfer Protocol Secure (HTTPS) *(171)*
IPSec *(171)*
key escrow *(156)*
key exchange *(155)*
key stretching *(157)*
Lightweight Directory Access Protocol Secure (LDAPS) *(170)*
meet-in-the-middle attack *(176)*
message integrity *(149)*
Password-Based Key Derivation Function 2 (PBKDF2) *(157)*
Pretty Good Privacy (PGP) *(165)*
rainbow tables *(177)*
replay attack *(176)*
Secure IMAP *(171)*
Secure POP3 *(171)*
Secure/Multipurpose Internet Mail Extensions (S/MIME) *(170)*
Secure Real-time Transport Protocol (SRTP) *(170)*
Secure Shell (SSH) *(170)*
session key *(156)*

SFTP *(170)*
Simple Network Management Protocol version 3 (SNMPv3) *(171)*
steganography *(167)*
transport encryption *(157)*
Transport Layer Security (TLS) *(171)*

Key Terms Quiz

Use terms from the Key Terms list to complete the sentences that follow. Don't use the same term more than once. Not all terms will be used.

1. _____ is a protocol used to secure DNS packets during transmission across a network.
2. A common encryption method designed to encrypt above the network layer, enabling secure sessions between hosts, is called _____.
3. _____ is the use of special encoding to hide messages within other messages.
4. _____ provide precomputed answers to a problem.
5. A _____ is a software library that implements cryptographic functions.
6. E-mails and their attachments can be secured using _____.
7. The use of multiple nearly identical messages can lead to the _____ cryptographic attack method.
8. The _____ is a network protocol for securely delivering audio and video over IP networks.
9. Reusing previous user input to bypass security is an example of a(n) _____ attack.
10. _____ is a popular encryption program that has the ability to encrypt and digitally sign e-mail and files.

Multiple-Choice Quiz

1. Which of the following is used to strengthen passwords from brute force attacks?
   - **A.** Bcrypt2
   - **B.** PBKDF2
   - **C.** DNSSEC
   - **D.** SSH-enabled logins
2. Why is LSB encoding the preferred method for steganography?
   - **A.** It uses much stronger encryption.
   - **B.** It applies a digital signature to the message.
   - **C.** It alters the picture the least amount possible.
   - **D.** It adds no additional entropy.
3. Transport Layer Security consists of which two protocols?
   - **A.** The TLS Record Protocol and TLS Handshake Protocol
   - **B.** The TLS Record Protocol and TLS Certificate Protocol
   - **C.** The TLS Certificate Protocol and TLS Handshake Protocol
   - **D.** The TLS Key Protocol and TLS Handshake Protocol
4. What is the advantage of using a crypto module?
   - **A.** Custom hardware adds key entropy.
   - **B.** It performs operations and maintains the key material in a physical or logical boundary.
   - **C.** It performs encryption much faster than general-purpose computing devices.
   - **D.** None of the above.
5. Which of the following is a detailed standard for creating and implementing security policies?
   - **A.** PKIX
   - **B.** ISO/IEC 27002
   - **C.** FIPS
   - **D.** X.509
6. Why does ECC work well on low-power devices?
   - **A.** Less entropy is needed for a given key strength.
   - **B.** Less computational power is needed for a given key strength.
   - **C.** Less memory is needed for a given key strength.
   - **D.** None of the above.

7. What makes a digitally signed message different from an encrypted message?
   - **A.** The digitally signed message has encryption protections for integrity and nonrepudiation.
   - **B.** A digitally signed message uses much stronger encryption and is harder to break.
   - **C.** The encrypted message only uses symmetric encryption.
   - **D.** There is no difference.
8. Which of the following is a secure e-mail standard?
   - **A.** POP3
   - **B.** IMAP
   - **C.** SMTP
   - **D.** S/MIME
9. Which of the following is not an advantage of TLS v1.3 over TLS v1.2 and earlier?
   - **A.** Removal of RC4
   - **B.** Reduction in round trips during handshakes
   - **C.** Use of AES
   - **D.** Restriction to AEAD ciphers
10. Transport Layer Security for HTTP uses what port to communicate?
   - **A.** 53
   - **B.** 80
   - **C.** 143
   - **D.** 443

Essay Quiz

1. Imagine you are a web developer for a small, locally owned business. Explain when using HTTP would be satisfactory, and why, and explain when you should use HTTPS, and why.
2. Explain in your own words how, by applying both asymmetric and symmetric encryption, your browser uses TLS to protect the privacy of the information passing between your browser and a web server.
3. It is well understood that asymmetric encryption consumes more computing resources than symmetric encryption. Explain how PGP uses both asymmetric and symmetric encryption to be both secure and efficient.

Lab Projects

Note that for these lab projects, it would be best to have a partner so that you can each have your own pair of public/private keys to confirm the operation of PGP.

• Lab Project 6.1

Load either a trial version of PGP or Gnu Privacy Guard (GPG). Install it and create a public/private key pair for yourself. Create a document using a word processor and encrypt it using the receiver's public key. Send it to a partner (or yourself), and then decrypt it using the corresponding private key.

• Lab Project 6.2

Create another document different from the one used in Lab Project 6.1. This time use your private key to digitally sign the document and send it to a partner (or yourself), who can then use the public key to confirm that it really is from you, the indicated sender.